

Fatec Praia Grande

Paulo R. T. Cândido

## NodeJS - parte 1

Criar uma pasta para o projeto, abrir janela de comando e posicionar na pasta

1) Instalar o gerenciador de pacotes YARN (substitui o npm)

**npm install --global yarn**

2) Criar projeto

**yarn init -y**

3) Instalar Typescript (superset do Javascript - tipado)

Auxilia o desenvolvimento, não necessário em produção.

Para instalar apenas em desenvolvimento se utiliza o parâmetro -D.

Fornecer tipagem auxiliando no desenvolvimento.

**yarn add typescript -D**

4) Instalar o framework express

**yarn add express**

5) Instalar os types (tipos) do express

Apenas em ambiente de desenvolvimento.

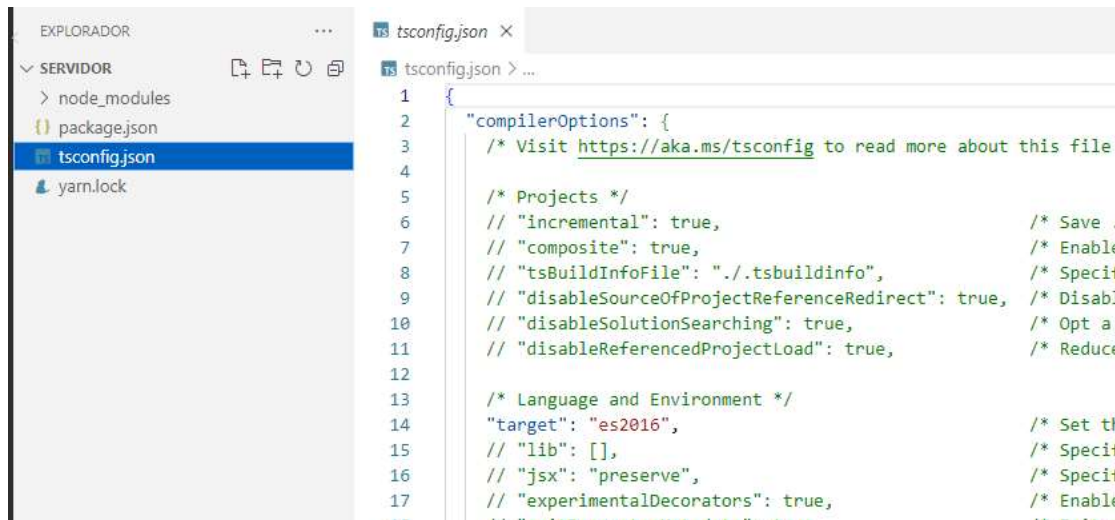
Fornecerá autocomplete para os comandos do express.

**yarn add @types/express -D**

6) Iniciar typescript no projeto

**yarn tsc --init**

Gera o arquivo tsconfig.json no projeto.



7) Instalar pacote necessário para criação de script (ver item 9 abaixo) para execução da aplicação com typescript.

**yarn add ts-node-dev -D**

8) Criar pasta src e nela arquivo server.ts para instanciação do express e configuração da aplicação.



Código:

```
// importa o express
import express from 'express';
// instancia o express
const app = express();
//configura porta e função executada na ativação
app.listen(4000, ()=>{console.log("Servidor Iniciado")});
```

9) Configurar script para iniciar a aplicação

```

{} package.json > ...
1  {
2    "name": "servidor",
3    "version": "1.0.0",
4    "main": "index.js",
5    "license": "MIT",
6    "scripts": {"dev": "ts-node-dev src/server.ts"},
7    "devDependencies": {
8      "@types/express": "^4.17.14",
9      "ts-node-dev": "^2.0.0",
10     "typescript": "^4.8.4"
11   },
12   "dependencies": {
13     "express": "^4.18.2"
14   }
15 }

```

10) Verificar se a aplicação inicia corretamente.

**yarn dev**

```

C:\Users\paulocandido\Documents\projetojs\servidor>yarn dev
yarn run v1.22.19
$ ts-node-dev src/server.ts
[INFO] 23:22:55 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.1, typescript ver. 4.8.4)
Servidor Iniciado

```

11) Definir rotas pelas quais as funcionalidades da aplicação são acionadas.

▼ SERVIDOR

- > node\_modules
- ▼ src
  - TS routes.ts
  - TS server.ts
  - { } package.json
  - tsconfig.json
  - yarn.lock

```

src > TS routes.ts > [0] default
1  // Importa componentes do express
2  import {Router,Request,Response} from 'express';
3
4  // Instancia roteador
5  const Roteador = Router();
6
7  // Define rota tipo get que, para funcionar, deve ser requisitada conforme exemplo.
8  // Exemplo de requisição: localhost:4000/teste/123?num=456
9  // Onde 123 e 456 podem ser substituídos por quaisquer valores
10 Roteador.get(
11   // URL com parâmetro :id
12   '/teste/:id',
13   // Função anônima com os parâmetros de tipos Request (requisição) e Response (resposta)
14   (req:Request,res:Response)=> {
15     // obtém query param
16     const x = req.query.num;
17     // obtém route param
18     const y = req.params.id;
19     res.send(`Resultado: ${Number(x) + Number(y)}`);
20   }
21 );
22 export default Roteador;

```

**Código:**

```

// Importa componentes do express
import {Router,Request,Response} from 'express';
// Instancia roteador
const Roteador = Router();
// Define rota tipo get que, para funcionar, deve ser requisitada conforme exemplo.

```

```
// Exemplo de requisição: localhost:4000/teste/123?num=456
// Onde 123 e 456 podem ser substituídos por quaisquer valores
Roteador.get(
  // URL com parâmetro :id
  '/teste/:id',
  // Função anônima com os parâmetros de tipos Request (requisição) e Response (resposta)
  (req:Request,res:Response)=> {
    // obtém query param
    const x = req.query.num;
    // obtem route param
    const y = req.params.id;
    return res.send(`Resultado: ${Number(x) + Number(y)}`);
  }
);
export default Roteador;
```

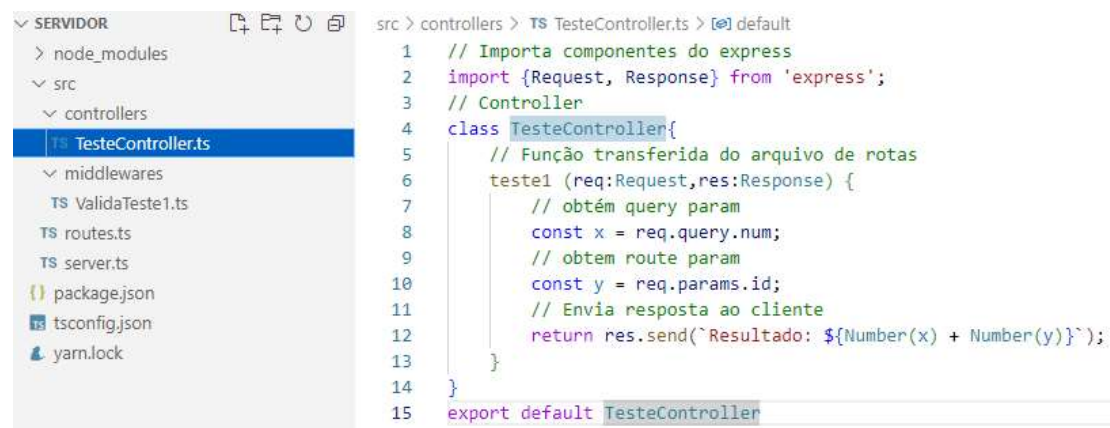
### Configuração de uso das rotas no arquivo server.ts

```
// importa o express
import express from 'express';
// importar rotas
import Roteador from './routes';
// instancia o express
const app = express();
// Configuração de uso das rotas
app.use(Roteador);
//configura porta e função executada na ativação
app.listen(4000, ()=>{console.log("Servidor Iniciado")});
```

## 12) Organizar retirando a função do arquivo de rotas e coloca-la em um controller

### routes.ts

```
// Importa componentes do express
import {Router} from 'express';
// Importa TesteController
import TesteController from './controllers/TesteController';
// Instancia roteador
const Roteador = Router();
// Define rota tipo get que, para funcionar, deve ser requisitada conforme exemplo.
// Exemplo de requisição: localhost:4000/teste/123?num=456
// Onde 123 e 456 podem ser substituídos por quaisquer valores
Roteador.get(
  // URL com parâmetro :id
  '/teste/:id',
  // Aciona função do TesteController
  new TesteController().teste1
);
export default Roteador;
```



```
src > controllers > TS TesteController.ts > [0] default
1 // Importa componentes do express
2 import {Request, Response} from 'express';
3 // Controller
4 class TesteController{
5     // Função transferida do arquivo de rotas
6     teste1 (req:Request,res:Response) {
7         // obtém query param
8         const x = req.query.num;
9         // obtém route param
10        const y = req.params.id;
11        // Envia resposta ao cliente
12        return res.send(`Resultado: ${Number(x) + Number(y)}`);
13    }
14 }
15 export default TesteController
```

```
// Importa componentes do express
import {Request, Response} from 'express';
// Controller
class TesteController{
    // Função transferida do arquivo de rotas
    teste1 (req:Request,res:Response) {
        // obtém query param
        const x = req.query.num;
        // obtém route param
        const y = req.params.id;
        // Envia resposta ao cliente
        return res.send(`Resultado: ${Number(x) + Number(y)}`);
    }
}
export default TesteController
```

### 13) Validar os parâmetros usando middleware

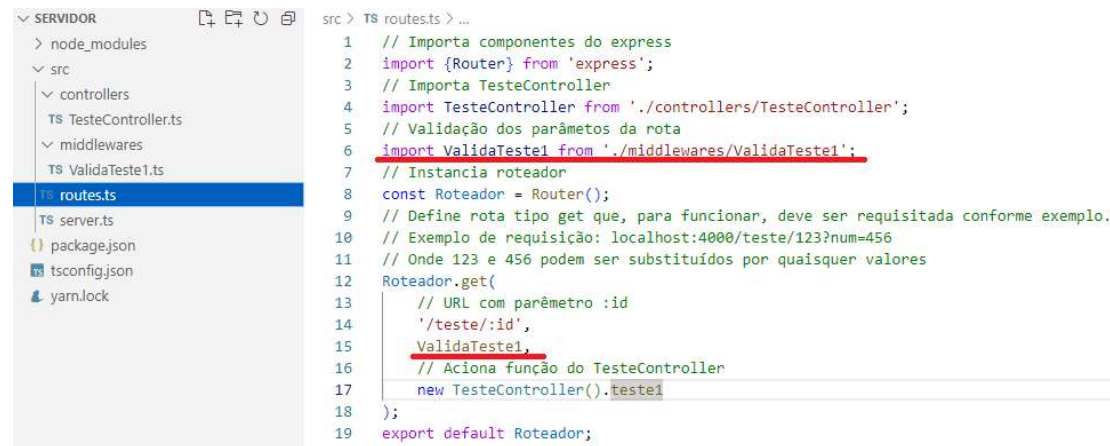


```
src > middlewares > TS ValidaTeste1.ts > ValidaTeste1
1 import {Request,Response,NextFunction} from 'express';
2 // verifica se os parâmetros da requisição são válidos
3 function ValidaTeste1 (req:Request,res:Response,next:NextFunction) {
4     const id = req.params.id;
5     const num = req.query.num;
6     if (Number(id)>1000 || num==null)
7     {
8         // Retorna código 400 indicando bad request
9         res.status(400).send("Parâmetros Inválidos");
10    }
11    // chama a próxima função na rota e retorna sua resposta
12    return next();
13 }
14 export default ValidaTeste1
```

```
import {Request,Response,NextFunction} from 'express';
// verifica se os parâmetros da requisição são válidos
function ValidaTeste1 (req:Request,res:Response,next:NextFunction) {
    const id = req.params.id;
    const num = req.query.num;
    if (Number(id)>1000 || num==null)
    {
        // Retorna código 400 indicando bad request
        res.status(400).send("Parâmetros Inválidos");
    }
    // chama a próxima função na rota e retorna sua resposta
    return next();
}
```

export default ValidaTeste1

## Usando o middleware na rota



```
src > TS routes.ts > ...
1 // Importa componentes do express
2 import {Router} from 'express';
3 // Importa TesteController
4 import TesteController from './controllers/TesteController';
5 // Validação dos parâmetros da rota
6 import ValidaTeste1 from './middlewares/ValidaTeste1';
7 // Instancia roteador
8 const Roteador = Router();
9 // Define rota tipo get que, para funcionar, deve ser requisitada conforme exemplo.
10 // Exemplo de requisição: localhost:4000/teste/123?num=456
11 // Onde 123 e 456 podem ser substituídos por quaisquer valores
12 Roteador.get(
13   // URL com parâmetro :id
14   '/teste/:id',
15   ValidaTeste1,
16   // Aciona função do TesteController
17   new TesteController().teste1
18 );
19 export default Roteador;
```

```
// Importa componentes do express
import {Router} from 'express';
// Importa TesteController
import TesteController from './controllers/TesteController';
// Validação dos parâmetros da rota
import ValidaTeste1 from './middlewares/ValidaTeste1';
// Instancia roteador
const Roteador = Router();
// Define rota tipo get que, para funcionar, deve ser requisitada conforme exemplo.
// Exemplo de requisição: localhost:4000/teste/123?num=456
// Onde 123 e 456 podem ser substituídos por quaisquer valores
Roteador.get(
  // URL com parâmetro :id
  '/teste/:id',
  ValidaTeste1,
  // Aciona função do TesteController
  new TesteController().teste1
);
export default Roteador;
```